A supporting tool for requirements change management in distributed agile development

ramdan mowad
Faculty of Computers and Information Technology, Future University in Egypt, Cairo, Egypt, ramdan.mowad@fue.edu.eg

Domia Lloyd
College of Computing and Information Technology, Arab Academy for Science and Technology and Maritime Transport, Cairo, Egypt, domia4@yahoo.com

Mona Kadry
College of Computing and Information Technology, Arab Academy for Science and Technology and Maritime Transport, Cairo, Egypt, monakadry@yahoo.com

Follow this and additional works at: https://digitalcommons.aaru.edu.jo/fcij
Part of the Computational Engineering Commons

Recommended Citation
Available at: https://digitalcommons.aaru.edu.jo/fcij/vol2/iss1/1

This Article is brought to you for free and open access by Arab Journals Platform. It has been accepted for inclusion in Future Computing and Informatics Journal by an authorized editor. The journal is hosted on Digital Commons, an Elsevier platform. For more information, please contact rakan@aaru.edu.jo, marah@aaru.edu.jo, u.murad@aaru.edu.jo.
A supporting tool for requirements change management in distributed agile development

Domia Lloyd, Ramadan Moawad, Mona Kadry

Abstract

Software development industry has witnessed the growth of the agile movement and approaches. Applying the agile approaches and practices in the distributed environment will lead to gain a lot of benefits such as reduced costs, higher efficiency and better customization, on the other hand it will face many challenges for example working in different time zones, requirements changes, personal selection and knowledge management. In order to gain these benefits, it should address the challenges that will face the agile approaches in a distributed environment. One of the main challenges is managing the requirements changes during the process of distributed agile software development. Only few researches published in the literature, addressed the problem of requirements changes during the development process in distributed agile development. Most of the published researches in this context are based on industrial experiences which increases the need for combining the industry and academia within this area. In this paper an approach to manage requirements changes in distributed agile development is introduced. This suggested approach works to fill the gap between the industry and research in distributed agile development by combining the industrial practice and academic technique. This approach is based on a proposed feature model called a features tree. The approach is associated with a supporting software tool that helps to manage the requirement changes in distributed agile development. The supporting tool is tested and evaluated in real environments by software development professionals using an exhaustive set of criteria, and the results are promising.

Keywords: Distributed agile; Requirements changes

1. Introduction

One of the main challenges that face the process of software requirements is the communication between the onshore and offshore sites in distributed teams. To reduce iteration time at the offshore site requires more efficient communication with the onshore site. Previous knowledge in certain domain is very important to understand software requirements. Agile methods are considered to be a good solution when they work in small iterations to deliver complete software solution at the end of these iterations. The Agile development aims to increase the team effectiveness, by reducing time of exchanging and sharing information between people [1]. Another aspect of Agile is minimizing the delivered documentation.

In distributed agile, the development activities are divided into two categories: onshore activities and offshore activities [2]. One of the main reasons why the software projects are deployed at offshore is that by taking advantage of cheaper labor, facilities and talented workforce [2]. Reducing the development cost is one of the main advantages that can be achieved using offshore development [3].
The selection of Username feature in Login feature requires the selection of Username feature in Registration main feature, as well as the selection of Email feature requires the selection of Email feature in Registration main feature. The third and last main feature is authentication feature and it is an optional feature. Authentication feature contains two alternative features: Email Authentication and SMS authentication. The selection of the SMS feature requires the selection of Mobile No. feature and the selection of Email in Authentication requires the selection of Email in Registration feature. The feature model of web registration system can satisfy the requirements for more than one application in the domain of web registration and that what is called the molding requirements for a family of systems or product line.

The feature model is used to model variability for a family of systems or product line in a specific domain. In our case we won't do that. The aim of our approach is to model the requirements for single software application in distributed agile environment. The software requirements in this case are subject to change from time to time during the development process. We aim to adopt the feature model to manage the requirements’ changes and represent these changes as variability to study the impact of these changes to the other requirements which is called requirements dependency. On the other hand modeling requirements changes will help the offshore development team to understand the new requirements and the possibility of development. In the next section we will introduce our suggested feature tree which contains some modifications to the feature model to support the process of managing requirements changes in the distributed agile development.

The adoption process of feature model will include new notations. All requirements will be modeled as a mandatory feature with considerable concentration on the relation between requirements which is called requirements dependency in the form of Implication and Exclusion relationship. There is no need to indicate the optional, mandatory or alternative requirements which is used in product line. To manipulate and manage requirements’ changes the authors suggest new notation to represent the processes of adding, updating and deleting requirements. Table 1 presents the suggested notations to maintain and control the requirements’ changes processes which will be represented in the feature tree in next section.

3. Features tree

The process of constructing the feature tree consists of two main steps. The first step aims to construct the initial form of the feature tree. The initial form of features tree will contain all candidate requirements. Fig. 2 shows an example of the initial feature tree.

The second step is managing the requirements changes. After each iteration the development team will take the decision of approving the new changes or preventing it using the approval and prevention notations which presented in Table 1. The change management process can be summarized in three activities. a) Modification of requirements in the form of...
adding, deleting, and updating. b) Controlling of changes in the form of approve or prevent. c) Keep tacking of dependency relationship between requirements in the form of implication and exclusion.

4. Features tree in distributed agile

To introduce a complete distributed agile approach with requirement change management, the authors combined the suggested feature tree with The New Agile Process for Distributed Projects (NAPDP) [8,9]. NAPDP is an approach to handle Global Agile Development (GAD); it provides benefits to the domain knowledge exchange and proposes a better branching and release management. It also provides a set of best practices, helping to link design, test approaches and tools, which help to maintain a software architecture.

NAPDP is based on exploring different approaches which follow the agile development models and the agile manifesto.
NAPDP has combined the most appropriate methods and best practices from Rational Unified Process (RUP) [12,13], Extreme Programming (XP) [10], Scrum and Rapid Object Oriented Process for Embedded System (ROPES) [11]. The suggested approach has taken the advantages of NAPDP and combined these advantages with the suggested feature tree. The proposed approach consists of two main phases: Project Initialization and Development Cycle, each phase contains several steps.

**Project initialization phase**

During the project initialization phase the project manager and requirements engineers capture requirements from the customer/stakeholders. Many planning activities are also conducted in this phase. This phase consists of the following three steps:

1) **Project planning**

At the earlier project meetings, several activities are conducted: project planning and scheduling, allocation of resources, development strategies including development plan and configuration management.

2) **Requirements Engineering**

In this step requirement engineers capture requirements from different resources such as domain experts and stakeholders. The requirements engineers are also responsible for eliciting, analyzing, specifying, validating requirements and building the initial features tree.

3) **Creating the Prioritized Feature Lists**

In this step the Prioritized Feature Lists are created. Each list will be implemented in a development cycle iteration and integrated into the system. Project manager, software architect, test and quality manager, feature designer, requirements engineers, configuration manager and executive programmers, all are involved in this step.

**Development cycle phase**

In development cycle phase a certain number of features are implemented. During each development cycle iteration the detailed feature tree is maintained. Features are designed, implemented, and tested. A quick prototype of the implemented features is delivered for customer evaluation at the end of each iteration. In the following section we will explain each step in more details.

1) **Analysis**

Each development cycle iteration starts with analysis step. This step includes requirement analysis, architecture analysis and object analysis.

2) **Maintaining Feature Tree**

The process of maintaining feature tree includes adding new features, modifying or updating existing features and deleting existing features. The maintenance process includes also the approval of the requirement change and the tracking of dependency relationship between requirements.
3) Design

After the analysis is completed and the new requirements changes are maintained in the feature tree, the process goes to the design step. This step starts from architectural design.

4) Implementation

The implementation step translates features into achievable solution. The developers write the required code to develop the features.
5) Test

The test phase assures that the features are already designed and implemented correctly. This step starts after all developers completing the process of writing codes. Test results are evaluated and sent back to re-implementation in case of test failure.

6) Advance

In this step the customers are involved with development team to evaluate the developed features. The development team can subsequently review the codes and edit it if needed based on the customers' comments. Fig. 3 shows the phases of the suggested approach.

5. The supporting tool

To help in applying the suggested approach a supporting tool is designed. It concentrates on managing the requirements changes in distributed agile development. The tool takes into consideration the nature of distributed agile such as the different geographical locations of the development teams, communication between development team and knowledge management. It is a web based tool intended to facilitate and manage the requirements modeling and changes in all phases. It starts capturing requirements from the project initiation phase and keeps track of requirements and its changes during all next steps and iterations. The tool also allows the project manager to keep track of the status of features in each
development step. Fig. 4 represents the activity diagram of the supporting tool.

Project manager works closely with different parties of stakeholders and development teams to identify the development members, and then the project manager assigns each member to a team. After that the requirements engineers create the initial form of the feature tree, constraints among features are also defined by them. Project manager is involved again in creating the prioritized feature list, each feature list will be assigned to one team. The distributed teams will be involved to record each finished features from the prioritized list in each development cycle step starting from analysis step. The development teams also maintain and record the requirements changes into the feature tree. The development teams keep recording finished features in all steps and the project manager will be able to monitor and track the finished features at each step with the progress of the project. The final step is the acceptance which conducted closely with the customer. Fig. 5 demonstrates how the supporting tool can maintain feature tree.

6. Evaluating the supporting tool

To evaluate this work, we have contacted several software development organizations in Egypt who might be interested in the area of distributed agile development. One of these organizations is the IT department of the National Research Center (NRC) in Egypt. The main responsibilities of the IT
department are developing, maintaining, replacing, and upgrading the software systems for the NRC. NRC is the largest research center in Egypt which has more than 4000 researchers working in different trends of research. The IT department at The NRC showed an interest for evaluating and validating the supporting tool since there is an increasing need for working in distributed development environment. They were about to develop a Document Management System (DMS), the system goal is to initiate, track, manage and store documents in order to reduce paper work. The development teams are located in different building and branches of the NRC; each team is responsible for covering and satisfying the software needs of certain sectors.

Evaluation criteria

After conducting several meetings with the project manager and the team leaders to identify the evaluation criteria, they agreed for these evaluation criteria: concept, design, usability, capabilities, security, traceability, accessibility, multi-user, configurations, generality and tasks assignment [14,15]. Table 2 presents the evaluation criteria with more explanation.

Evaluation teams responsibilities

The responsibilities of the evaluation teams are:

- Elicit and collect requirements from different users.
- Identify the feature of requirements, or translate the user requirements into features.
- Build the feature tree.
- Initiate constraints between features.
- Assign features to different development team; each set of features will form an iteration.
- Estimate the time required for each iteration, according to the project manager vision and the complexity of the features.
- Review and maintain requirements in each iteration, and if needed update the features tree.
- Maintain the relation between features at the development cycle phase.
- Keep track with finished feature at each step in the development phases.

Findings

Table 3 introduces the finding of the tool according to development teams’ evaluation.

Recommendations

Table 4 presents the recommendations for the tool which will increase the functionality and usability of the tool in future.

Table 4

Recommendations for the supporting tool.

<table>
<thead>
<tr>
<th>No.</th>
<th>Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Concept</td>
<td>• The tool needs to extend its functionality to consider assigning tasks to the development member.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The tool needs to add some reports to measure the performance of the development team.</td>
</tr>
<tr>
<td>2</td>
<td>Design</td>
<td>• The development teams prefer to show the web application messages in popup windows rather than a text on the web page with red color.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The development teams asked to add favorite menu, which will contain the frequently visited pages of the web application.</td>
</tr>
<tr>
<td>3</td>
<td>Usability</td>
<td>• The tool won’t be usable without the knowledge of the approach.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The tool available only on English and it will be more usable if it supported more languages.</td>
</tr>
<tr>
<td>4</td>
<td>Capabilities</td>
<td>• Extend the functionality of the tool to consider assigning tasks to the development member.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The tool should notify the project manager in case of the changes in the feature tree.</td>
</tr>
<tr>
<td>5</td>
<td>Security</td>
<td>• The tool should notify the users with their security levels.</td>
</tr>
<tr>
<td>6</td>
<td>Tractability</td>
<td>• The tool doesn’t have graphical figures to represent the progress of the project.</td>
</tr>
<tr>
<td>7</td>
<td>Generality</td>
<td>• The tool approved its ability to manage the requirements changes in the context of distributed agile development for database applications.</td>
</tr>
<tr>
<td>8</td>
<td>Tasks assignment</td>
<td>• The tool doesn’t provide capability to allow the team leaders to assign takes to the development member.</td>
</tr>
</tbody>
</table>
7. Conclusion and future work

This paper studied different approaches and practices for GAD. The authors discovered that there is a lack for managing requirements changes practices and approaches in GAD. The majority of the existing research in the literature, are in the form of industrial experience reports. This paper has combined the feature model with an industrial approach in order to provide a solution that can manage requirements in GAD. The authors have modified the feature model to handle the requirements changes process. The modified feature model used to model requirements for single system and manage its changes, while the original feature model is used mainly in modeling a family of systems or software product line. The modified feature model is called here a features tree. The features tree is integrated with NAPDP which is an approach for distributed agile development. NAPDP consists of two phases, project initiation phase and development cycle phase. Managing requirements changes take place in the two phases, it starts with a project initiation phase and continue through a development cycle phase. This approach is associated with a supporting tool to aid its theoretical vision. The tool is a web based tool to support the distributed agile development. The tool helps to manage the requirements changes and helps the project manager to keep track of project status at the two phases. The supporting tool is evaluated in a real environment at NRC. The project used in the evaluation was a document management system which is medium size database system. The findings of the evaluation are encouraging since they are based on exhaustive set of criteria developed by professional software developers.

In the future, the authors are going to use the tool in different projects to enhance its performance and to provide it with more functionalities. Applying the tool evaluation recommendations will enhance the usability of the tool, and will make it able to be used in many software projects with different sizes and domains.

References